

Comptes Unix stockés sur LDAP

Stéphane Bortzmeyer

<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 26 Mars 2008. Dernière mise à jour le 30 Novembre 2008

<http://www.bortzmeyer.org/comptes-unix-ldap.html>

Si on gère de nombreux ordinateurs multi-utilisateurs, on se pose forcément la question de la centralisation des **comptes**, des informations sur les utilisateurs. Il est très pénible de devoir créer un compte sur N machines dès qu'un nouvel employé arrive ou de devoir se rappeler des M mots de passe précédents parce qu'on ne les a pas changés sur toutes les machines. Il existe plusieurs solutions techniques à ce problème, et je présente ici l'utilisation de LDAP pour gérer une base centralisée de comptes, notamment pour des machines Unix.

LDAP est un protocole client-serveur d'interrogation d'un annuaire. Il offre donc quelques analogies avec le DNS. Mais son histoire est très différente. Issu à l'origine du projet X.500, il en est aujourd'hui le seul survivant. LDAP est normalisé dans les RFC 4510¹ et suivants.

LDAP est juste un protocole, il n'impose rien sur le mécanisme de stockage des données, dont on dira un mot au moment de configurer le **serveur**. Mais nous allons commencer par la configuration du **client**, car il y a davantage de clients que de serveurs.

Nous avons donc une machine Unix (en l'occurrence une Debian) qui a une base d'utilisateurs traditionnelle (le fichier `/etc/passwd`) et qui voudrait l'enrichir avec les informations venues du serveur LDAP `ldap.example.org`. L'administrateur de ce serveur nous a donné le nom du serveur et la **base** à utiliser, `dc=example,dc=org` (contrairement au DNS, LDAP ne part pas de la racine mais d'une base, souvent dérivée d'un nom de domaine). L'administrateur nous dira également la version utilisée (la version 3 reste aujourd'hui la seule) et le mécanisme d'authentification auprès du serveur (contrairement au DNS, un serveur LDAP peut nécessiter une authentification des clients), l'usage ou non de TLS, etc. Les objets LDAP décrivant les utilisateurs seront probablement de la **classe** `posixAccount`.

1. Pour voir le RFC de numéro NNN, <http://www.ietf.org/rfc/rfcNNN.txt>, par exemple <http://www.ietf.org/rfc/rfc4510.txt>

Que faut-il connaître sur LDAP encore ? Outre les références à la fin de cet article, nécessaires pour tout savoir, notons simplement que chaque objet stocké dans la base LDAP a un DN ("*Distinguished Name*") et que ce DN l'identifie de manière unique. Le DN est formé de plusieurs composants, chaque composant étant un doublet attribut=valeur. Par exemple, `uid=bleriot,ou=People,dc=example,dc=org` est un DN.

Utilisons le client `ldapsearch` (un outil de test, comme `dig` pour le DNS ; sur Debian, il est dans le paquetage `ldap-utils`) pour tester que notre machine peut joindre le serveur :

```
% ldapsearch -h ldap.example.org -x -b dc=example,dc=org \
      '(objectClass=posixAccount)'
...
dn: cn=Louis Bleriot,ou=People,dc=example,dc=org
objectClass: posixAccount
objectClass: shadowAccount
cn: Louis Bleriot
sn: bleriot
uid: bleriot
uidNumber: 1011
homeDirectory: /home/bleriot
gecos: Louis Bleriot
loginShell: /usr/bin/zsh
...
```

(`-x` veut dire accès anonyme, sans authentification) Une fois qu'on est sûr du serveur, nous pouvons commencer à configurer le client.

Pour éviter de mettre du code LDAP dans chaque application qui fait de l'authentification (une liste pas du tout limitative nous donne `sshd`, `login`, `sudo`, etc), Unix utilise en général le système PAM dans lequel l'application appelle un greffon PAM pour les différentes opérations. PAM à son tour charge, selon sa configuration, du code LDAP (ou bien utilisant d'autres techniques d'authentification).

On va donc installer le support LDAP de PAM. Sur une Debian, c'est le paquetage `libpam-ldap`. Il faut le configurer pour lui donner les informations que nous a transmises l'administrateur système. Cela se fait dans `/etc/pam_ldap.conf` dont voici les extraits pertinents :

```
# On peut aussi indiquer le serveur par la directive "uri" mais je n'ai
# jamais réussi à la faire fonctionner
host ldap.example.org
base dc=example,dc=org
# Ne pas oublier ces trois lignes, commentées par défaut !
nss_base_passwd ou=People,dc=example,dc=org?one
nss_base_shadow ou=People,dc=example,dc=org?one
nss_base_group ou=Group,dc=example,dc=org?one
# Pour savoir quel couple attribut/valeur ajouter à la base (ici ou=People),
# demander à l'administrateur du serveur LDAP.
```

On configure ensuite PAM pour utiliser LDAP, dans `/etc/pam.d`. Chaque application qui authentifie a un fichier dans ce répertoire mais la plupart des fichiers se contentent d'inclure les fichiers communs comme `common-auth`. Éditions `common-auth` pour se servir de LDAP (l'ordre du fichier est significatif, c'est celui dans lequel on utilisera les différentes techniques) :

```
# Utiliser LDAP
auth sufficient pam_ldap.so
# Garder une authentification traditionnelle au cas où...
auth required pam_unix.so nullok_secure
```

Même chose pour `common-account` et `common-password`.

Sommes-nous prêts, désormais, à authentifier ? Pas tout à fait. PAM ne fait que l'authentification. Mais une base d'utilisateurs, sur Unix, sert à bien d'autres choses. Par exemple, même une commande aussi simple que `ls -l` va nécessiter de traduire les `uid` numériques que contient le fichier en noms alphabétiques. Unix fait pour cela appel à un autre système, dont la configuration doit se faire indépendamment de LDAP, NSS (le projet NSS a été décrit à l'origine dans le RFC 2307).

Installons donc NSS LDAP, c'est le paquetage Debian `libnss-ldap`. Puis configurons-le dans `/etc/libnss-ldap.conf`.

```
# On peut aussi indiquer le serveur par la directive "uri" mais je n'ai
# jamais réussi à la faire fonctionner
host ldap.example.org
base dc=example,dc=org
# rootbinddn ne fonctionne pas pour moi et n'est de toute façon pas
# très utile
#rootbinddn cn=admin,dc=example,dc=org
# Ne pas oublier ces trois lignes, commentées par défaut !
nss_base_passwd ou=People,dc=example,dc=org?one
nss_base_shadow ou=People,dc=example,dc=org?one
nss_base_group ou=Group,dc=example,dc=org?one
# Pour savoir quel couple attribut/valeur ajouter à la base (ici ou=People),
# demander à l'administrateur du serveur LDAP.
```

Puis indiquons à NSS qu'il doit utiliser LDAP, ce qui se fait dans `/etc/nsswitch.conf` :

```
passwd:          compat ldap
group:           compat ldap
shadow:         compat ldap
```

Maintenant, tout devrait marcher. Il est temps de tester. Commençons par demander la liste des comptes :

```
% getent passwd
...
bleriot:x:1011:110:Louis Bleriot, Test account:/home/bleriot:/usr/bin/zsh
```

On doit voir apparaître les comptes purement LDAP. On peut aussi utiliser des commandes qui affichent des informations sur les utilisateurs :

```
% finger bleriot
Login: bleriot                Name: Louis Bleriot
Directory: /home/bleriot     Shell: /usr/bin/zsh
Office: Test account
...
```

Ou bien (**attention**, après la modification de `nsswitch.conf`, les applications qui tournent doivent être redémarrées. `finger` était lancé après mais le shell a sans doute été lancé avant et c'est lui qui interprète le tilde) :

```
% cd ~bleriot
% pwd
/home/bleriot
```

Enfin, il ne reste qu'à s'authentifier (même avertissement : il faut redémarrer les programmes qui utilisent NSS comme le serveur SSH). Normalement, on doit pouvoir désormais se connecter et travailler comme si on avait un compte local.

Et si cela ne marche pas ? Il faut tester avec `ldapsearch`, regarder les journaux du serveur SSH et, si on y a accès, regarder le journal du serveur LDAP (`/var/log/syslog` par défaut sur Debian). On peut aussi se servir d'autres clients LDAP comme `ldapsh` [<http://ldapsh.sourceforge.net/>](http://ldapsh.sourceforge.net/) qui permet de naviguer dans la base LDAP comme si c'était un système de fichiers, ou comme l'excellent programme graphique `GQ` [<http://gq-project.org/>](http://gq-project.org/).

D'autres programmes peuvent s'authentifier avec LDAP. Par exemple, le SGBD PostgreSQL a une telle option [<http://www.postgresql.org/docs/current/static/auth-methods.html#AUTH-LDAP>](http://www.postgresql.org/docs/current/static/auth-methods.html#AUTH-LDAP) qu'on peut mettre dans son `pg_hba.conf`. Elle a l'air simple mais je ne suis jamais arrivé à obtenir de PostgreSQL des requêtes LDAP correctes, même après avoir beaucoup regardé ce que recevait le serveur LDAP. L'authentification LDAP de PostgreSQL est en outre assez rigide (pas de filtres, par exemple). Donc, j'utilise plutôt l'option PAM qui tire profit de la configuration ci-dessus :

```
hostssl all USERNAME 192.0.2.240/28 pam
```

Maintenant, si le système d'exploitation utilisé n'est pas Debian, mais FreeBSD ? Depuis quelques années, FreeBSD dispose également de PAM et de NSS. La démarche est la suivante (elle est bien documentée dans *"LDAP Authentication"* [<http://www.freebsd.org/doc/en/articles/ldap-auth/index.html>](http://www.freebsd.org/doc/en/articles/ldap-auth/index.html) :

- Installer les ports `pam_ldap` et `nss_ldap`.
- Configurer le client LDAP en `/usr/local/etc/openldap/ldap.conf`.
- Configurer PAM en `/etc/pam.d`.
- Configurer NSS en `/usr/local/etc/nss_ldap.conf` et `/etc/nsswitch.conf`.

Une fois les ports installés, il faut indiquer au client LDAP de la machine où trouver les informations. Voici un exemple de `/usr/local/etc/openldap/ldap.conf` :

```
BASE dc=example,dc=org
URI ldap://ldap.example.org/
pam_login_attribute uid
```

On a indiqué la base LDAP, l'URI du serveur LDAP et l'attribut LDAP qui servira de nom au compte Unix (`uid`).

Ensuite, il faut configurer PAM dans chaque application, via les fichiers en `/etc/pam.d`. Le port FreeBSD de PAM n'inclus pas de fichiers communs, hélas, et il faut donc configurer un fichier par application (peut-être peut-on les réécrire avec des directives incluant un fichier commun, je n'ai pas essayé). Par exemple, pour SSH, on édite `/etc/pam.d/ssh` ainsi :

```
auth sufficient /usr/local/lib/pam_ldap.so no_warn
...
auth required pam_unix.so no_warn try_first_pass

account required /usr/local/lib/pam_ldap.so no_warn ignore_authinfo_unavail ignore_unknown
...
account required pam_unix.so
```

<http://www.bortzmeyer.org/comptes-unix-ldap.html>

Maintenant, on peut se connecter mais, pour avoir tous les services de noms, il faut NSS. On édite `/usr/local/etc/nss_ldap.conf` ainsi :

```
host ldap.example.org
base dc=example,dc=org
nss_base_passwd ou=People,dc=example,dc=org?one
nss_base_shadow ou=People,dc=example,dc=org?one
nss_base_group      ou=Group,dc=example,dc=org?one
```

Et il faut indiquer à la libc d'utiliser LDAP en modifiant `/etc/nsswitch.conf` :

```
group: files ldap
passwd: files ldap
```

Désormais, tout marche, notre machine FreeBSD peut utiliser tous les services de noms que fournit LDAP.

Et sur le serveur ? Si on doit le configurer, s'il n'a pas déjà été fait, voici les étapes.

On installe le logiciel serveur d'OpenLDAP (paquetage `slapd`). Pour sa configuration, on édite `/etc/ldap/slapd.conf` notamment pour y mettre :

```
suffix          "dc=example,dc=org"
```

et s'assurer que les permissions (directives `access`) comportent bien notre base.

Si on a un schéma LDAP spécifique, il faut aussi le mettre dans ce fichier :

```
include          /etc/ldap/schema/example.schema
```

On relance le serveur, on vérifie qu'il tourne et qu'on peut l'interroger avec `ldapsearch`.

Sa base est actuellement vide, il nous faut donc créer des comptes. Il existe plusieurs solutions, le choix dépend des circonstances.

- Si on a une liste d'utilisateurs dans une autre base de données, on peut écrire un programme qui exporte cette base en format LDIF et charger ensuite le fichier LDIF, serveur arrêté, avec `slapadd < example.ldif`.
- Si on veut procéder entrée par entrée, on peut aussi utiliser un fichier LDIF (comme celui-ci (en ligne sur <http://www.bortzmeyer.org/files/yet-another-account.ldif>)) et le charger via le protocole LDAP, avec `ldapadd -W -D 'cn=admin,dc=example,dc=org' -f yet-another-account.ldif`, ce qui ne nécessite pas d'arrêter le serveur et permet de travailler depuis une autre machine.

<http://www.bortzmeyer.org/comptes-unix-ldap.html>

– On peut aussi utiliser un programme local de création de compte comme ce petit programme (en ligne sur <http://www.bortzmeyer.org/files/adduser.py>) écrit en Python, avec la bibliothèque Python-LDAP <<http://python-ldap.sourceforge.net>>. Ce programme dépend d'un fichier de configuration (voici un exemple (en ligne sur <http://www.bortzmeyer.org/files/ldapsimpletools.ini>)) et s'utilise en tapant simplement `adduser USERNAME` (il demandera ensuite les autres informations nécessaires). Il n'existe pas de tel programme « standard » car chaque site a son propre modèle de données et ses propres règles. Une fois le ou les comptes créés, il faut évidemment tester que les clients LDAP puissent s'en servir, comme indiqué plus haut, lors de la configuration du client.

Détruire un compte peut également se faire de plusieurs façons :

Avec `ldapdelete` en indiquant le DN : `ldapdelete -w -D 'cn=admin,dc=example,dc=org' -x 'cn=Louis Bleriot,ou=People,dc=example,dc=org'`.

Avec `adduser` et son option `-r` (comme "remove"). Il peut être pratique, notamment pour déboguer des problèmes avec un client récalcitrant, de demander au serveur d'écrire dans son journal toutes les requêtes LDAP reçues. Cela ralentit le serveur, cela peut être très indiscret, mais c'est un bon outil de test. Il faut pour cela modifier la directive `loglevel` de `slapd.conf` :

```
# Read slapd.conf(5) for possible values
loglevel          256
```

On trouvera alors dans le journal des informations très indiscrettes comme (au moment d'une connexion SSH de l'utilisateur `bleriot`) :

```
Mar 26 10:15:05 lilith slapd[15932]: conn=410 op=1
      SRCH base="ou=People,dc=example,dc=org"
      scope=1 deref=0 filter="(uid=bleriot)"
```

(Un petit mot sur le journal : comme chaque requête LDAP est notée, cela peut ralentir sérieusement le serveur. Vérifiez bien dans la configuration de `syslog` que l'écriture est asynchrone, en mettant un tiret devant le nom du fichier journal.)

N'oubliez pas les sauvegardes ! La perte de la base de données du serveur LDAP peut être catastrophique. Les données étant stockées par OpenLDAP dans un fichier binaire, il est prudent de les sauvegarder en mode texte, avec le format LDIF. On peut copier toute la base avec `slapcat`.

Une fois les sauvegardes faites et vérifiées, vous pouvez aussi vous préoccuper des performances du serveur LDAP. `echoping` <<http://echoping.sourceforge.net>> a un greffon LDAP qui permet de tester les performances dudit serveur :

```
% echoping -n 5 -m ldap ldap.example.org \
  -b ou=People,dc=example,dc=org -s one -r '(uid=bleriot)'
Elapsed time: 0.000606 seconds
Elapsed time: 0.000493 seconds
Elapsed time: 0.000446 seconds
Elapsed time: 0.000425 seconds
Elapsed time: 0.000393 seconds
---
Minimum time: 0.000393 seconds (651399 bytes per sec.)
Maximum time: 0.000606 seconds (422442 bytes per sec.)
Average time: 0.000472 seconds (542373 bytes per sec.)
Standard deviation: 0.000074
Median time: 0.000446 seconds (573991 bytes per sec.)
```

Les utilisateurs d'Ubuntu apprécieront sans doute une documentation équivalente en <http://doc.ubuntu-fr.org/ldap_client>. Terminons par une bonne référence. Le Comité Réseau des Universités a une excellente page sur LDAP <<http://www.cru.fr/documentation/ldap/index>>, contenant des pointeurs vers toutes les documentations nécessaires.